TEC | Tecnológico de Costa Rica

Tarea No. 5

Eficiencia

Marco Rodríguez
Maximilian Latysh

Profesor Édgar Rojas Muñoz

Escuela de Computación
Ingeniería en Computación

Introducción a la Programación Grupo 21
IC1802

28 de abril de 2022

1. 5.5.1 Programando y probando el coeficiente binomial

- Función uno:

```
def f1(n, k):
   if k == 0 or n == k:
      return 1
   return funcion1(n - 1, k) + funcion1(n - 1, k - 1)
```

- Función dos:

```
def f2(n, k):
   if k == 0 or n == k: return 1
   suma = 0
   for i in range(k + 1):
      suma += f2(n - i - 1, k - i)
   return suma
```

- Función tres:

```
def factorial(n):
   mult = 1
   for i in range(2, n + 1):
      mult *= i
   return mult

def f3(n, k):
   return factorial(n)/(factorial(k)*factorial(n - k))
```

- Función cuatro:

```
def f4(n, k):
   if k == 0 or n == k:
      return 1
   return (n - k + 1)*f4(n, k - 1)/k
```

- Función cinco:

```
def f5(n, k):
   return f5aux(n, k, 1)

def f5aux(n, k, X):
   if k == 0 or n == k:
      return X
   return f5aux(n, k - 1, (n - k + 1)*X/k)
```

2. 5.5.3 Análisis

1. $O(n) = n$; Observamos que hay un solo ciclo en el cual el cambio se representa con una suma constante para cada iteración.

2. $O(n) = n^{\frac{1}{2}}$; Crecimiento fractorial. Hay un ciclo en el cual el cambio es una suma constante por cada iteración con la diferencia de que la comparación se hace con el cuadrado de i. Por lo tanto, se ejecutan $n^{\frac{1}{2}}$ pasos para que $i * i$ llega a ser más grande o igual que n.

3. $O(n) = n * log_2(n)$; Hay dos ciclos: en el primero, la variable se cambia con una suma constante por cada iteración; en el segundo, la variable se divida entre dos por cada iteración. Por lo tanto, la primera tiene crecimiento n y la segunda tiene crecimiento $log_2(n)$, y la combinación de los dos proporciona $n * log(n)$.

4. $O(n) = n^2$; Hay dos ciclos y ambos cambian por una variable constante por cada iteración respectiva. Por lo tanto, ambos son de crecimiento n, y la combinación de los dos proporciona n^2.

5. $O(n) = 3^n$; Por cada llamada de la función $f5(n)$, la cantidad de llamadas se extiende por 3. Por lo tanto, si n es un valor bajo da 3^1 (porque se tiene que llamar 3 ciclos). Si n es más grande, le toca llamar potencias de tres subsecuentemente.

3. 5.5.4 Mirando atrás

1. MCD: $O(n) = n^{\frac{1}{2}}$; entonces podemos interpretar el menor de a y b como el n. A partir de esto se puede concluir que la función raíz entera (Se basa en búsqueda binaria, tiene un $O(n)$ de $log^2(n)$). Después se utiliza el resultado de esa función (raíz de n) para hacer un ciclo que tiene un cambio representado por una suma constante por cada iteración pero como la iteración pasa por una raíz de n veces podemos concluir que el $O(n)$ de este ciclo es raíz n y por lo tanto el crecimiento de la función en si es raíz de n.

2. mcm: $O(n) = n^{\frac{1}{2}}$; podemos notar que el primer, tercer y cuarto ciclo tienen crecimientos que son necesariamente más pequeños que el segundo ciclo porque la cantidad de factores de un número es necesariamente más pequeña que la cantidad de pasos que se necesitan para calcular esos factores. Entonces en el análisis de la función el peor caso es cuando n es un número primo y en este caso duraría la raíz de $n/2$ pasos para llegar a la respuesta por lo tanto el $O(n)$ de este ciclo es la raíz de n.

3. Suma de cubos: $O(n) = n$; se puede notar qué hay un solo ciclo que tiene un cambio representado por una suma constante por cada iteración.

4. Es primo?: $O(n) = n^{\frac{1}{2}}$; podemos notar que el primer ciclo tiene un crecimiento que es necesariamente más pequeño que el segundo ciclo porque la cantidad de factores de un número es necesariamente más pequeña que la cantidad de pasos que se necesitan para calcular esos factores. Entonces en el análisis de la función el peor caso es cuando n es un número primo y en este caso duraría la raíz de $n/2$ pasos para llegar a la respuesta por lo tanto el $O(n)$ de este ciclo es la raíz de n.

5. n-simo primo: $O(n) = n * log(n)$; La función generar criba tiene dos ciclos principales: el ciclo central y el ciclo de marcado. La combinación de estos dos ciclos proporcionan $n * log(n)$ porque la cantidad de campos que el proceso de marcado tiene que llenar se incrementa cada vez.

6. Cantidad de dígitos: $O(n) = log(n)$ (base 10 naturalmente); el n se divide entre un valor constante por cada iteración hasta que llega a ser más pequeño que uno, por lo tanto es un crecimiento $log(n)$.

7. Invertir un número: $O(n) = log(n)$; el n se divide entre un valor constante por cada iteración hasta que llega a ser más pequeño que uno, por lo tanto es un crecimiento $log(n)$.

8. Factorial: $O(n) = n$; se tiene a un ciclo con un cambio representado por una suma constante por cada iteración.

9. Suma de fracciones: $O(n) = n$; se tiene a un ciclo con un cambio representado por una suma constante por cada iteración.

10. Suma de suma: $O(n) = n^2$; se nota que se tienen dos ciclos y ambos tienen un cambio representado por una suma constante por cada iteración.

11. Producto: $O(n) = n^2$; se tiene a un ciclo con un cambio representado por una suma constante por cada iteración pero que va hasta n^2.

12. Producto de sumas: $O(n) = n^3$; se tienen dos ciclos, el primer ciclo crece de una forma constante representado por una suma mientras que el segundo ciclo crece de la misma forma pero hasta el cuadrado de i que se generalizan como la suma desde 0 hasta n de $i * *2$.

13. Números perfectos: $O(n) = n$; se nota que en el ciclo más grande el crecimiento tiende a ser lineal.

14. Números amigos: $O(n) = n$; Se nota que en el ciclo más grande el crecimiento tiende a ser lineal (la función factores tiene crecimiento constante por suma).

15. Encontrar amigo: $O(n) = n$; Se nota que en el ciclo más grande el crecimiento tiende a ser lineal (la función factores, tiene crecimiento constante por suma).

16. Conjetura de Goldbach: $O(n) = n^{\frac{3}{2}}$; Se nota que la función de es primo tiene un $O(n)$ de raíz de n porque solo va hasta este punto en el ciclo principal (se puede ignorar el ciclo secundario). Pero, en la función $goldbach_r()$ el ciclo podría durar hasta un poco menos de $n/2$. Combinando estos dos, determinamos que el crecimiento proporcionada por los dos nos da $n^{3/2}$.

17. Cuatro cuadrados: $O(n) = n^2$; Primero se tiene que observar a la función raíz entera, la cual se basa en la búsqueda binaria. Por lo tanto, tiene un $O(n)$ de $log(n)$. Pero después, se tiene a 4 ciclos for en pila que van hasta la raíz de n proporcionada por la función establecida. De esta manera, se tiene que el $O(n) = (n^{1/2})^4 = n^2$.

18. Teorema de Carmichael: $O(n) = n * log(n)$; Raíz natura se cuenta con un $O(n) = log_2(n)$ y generar criba tiene uno de $O(n) = n * log(n)$. La criba supera al caso y proporciona $O(n) = n * log(n)$ como se había establecido antes.

Referencias

[1] E. R. Jiménez, *Fundamentos de Programación*. 2022.